Atlântico Avanti

Equipe 1 - Segmentação de vasos sanguíneos da retina

Alunos:

Felipe Cordeiro
Kauã Ribeiro
Fernanda Rocha
Luís Lima
João Vitor Lobo
Rodrigo Pinheiro

Sumário

- Introdução – Fernanda
- Metodologia – Felipe/João Vitor
- Resultados – Kauã
- Conclusões – Luís

Introdução

- Leitura de retina;
- Retinopatia diabética, retinopatia hipertensiva, degeneração macular relacionada à idade (DMRI), descolamento da retina e glaucoma;
- U-net.

Metodologia

Dataset

Pré - processamento de dados

Arquitetura U-NET

Treinamento e validação do modelo

Resultados

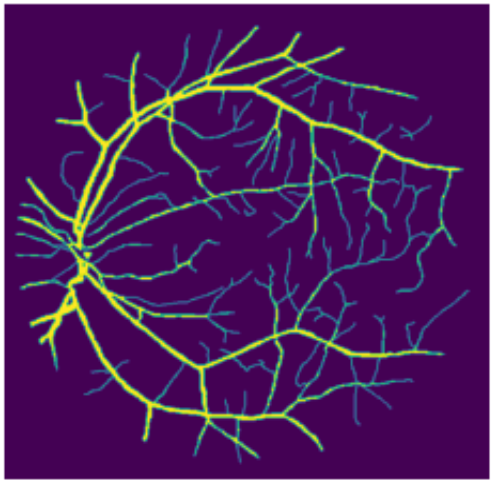
Atlântico Avanti

Dataset

Dataset

- Kaggle – Retina Blood Vessel
- 80 Imagens e 80 Máscaras
- Imagens: 512x512

49.png



49.png

Dataset – Setup de execução

- Input de imagens e mascaras

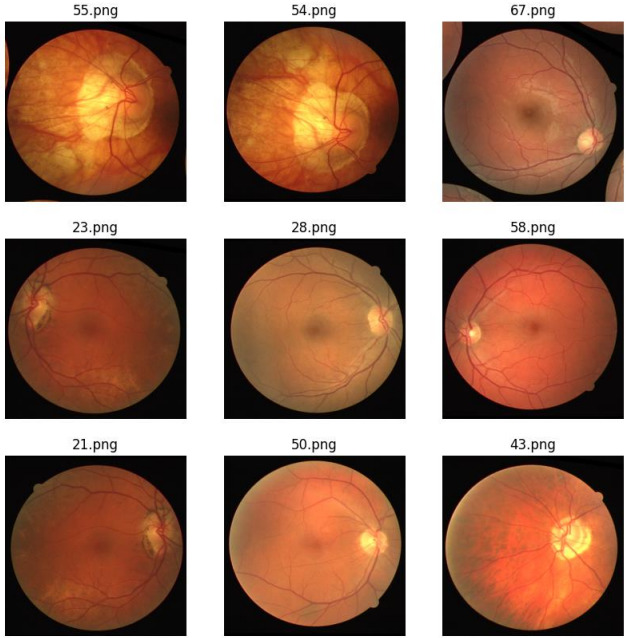
```
image_dir = '/content/drive/MyDrive/Bootcamp Atlantico/Data/train/image'
mask_dir = '/content/drive/MyDrive/Bootcamp Atlantico/Data/train/mask'
```

- Verificação da integridade das imagens

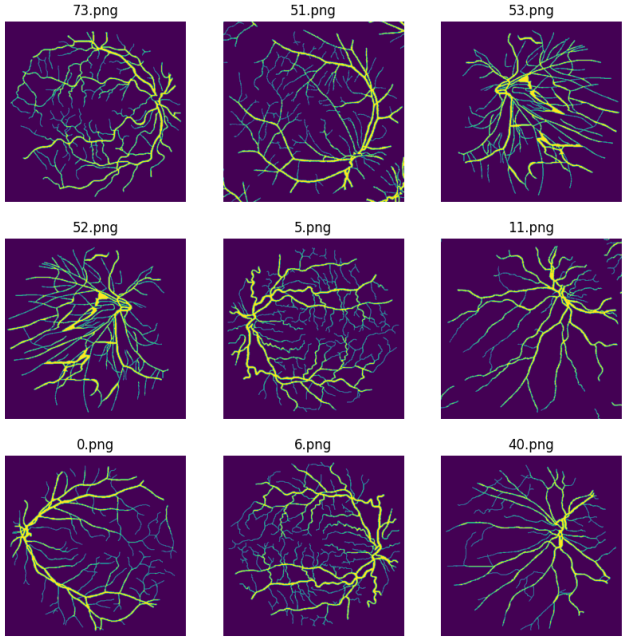
```
image_listdir = os.listdir(image_dir)
mask_listdir = os.listdir(mask_dir)
random_images = np.random.choice(image_listdir, size = 9, replace = False)
```

Dataset – Imagens

Vasos sanguíneos da retina



Mascara



Pré – Processamento

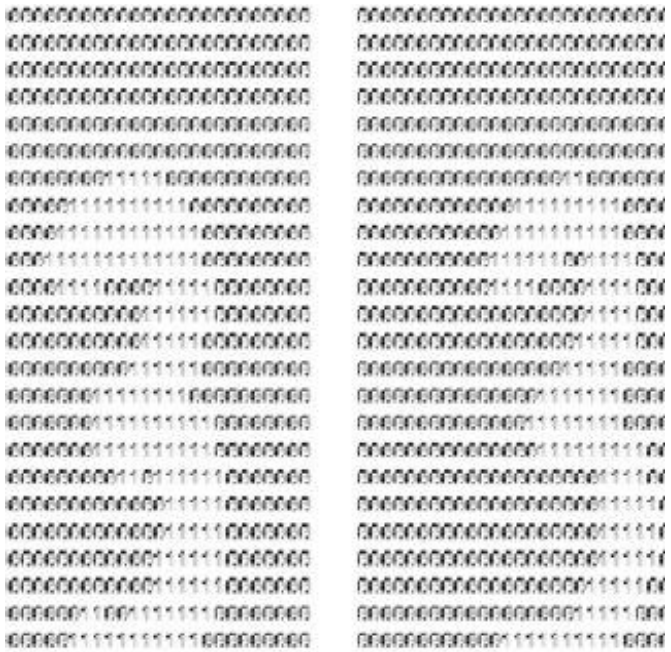
Normalização dos Dados

Normalização dos dados

```
#Normalize images
train_image_dataset = np.expand_dims(normalize(np.array(train_image_dataset), axis=1),3)
#D not normalize masks, just rescale to 0 to 1.
train_mask_dataset = np.expand_dims((np.array(train_mask_dataset)),3) /255.
```

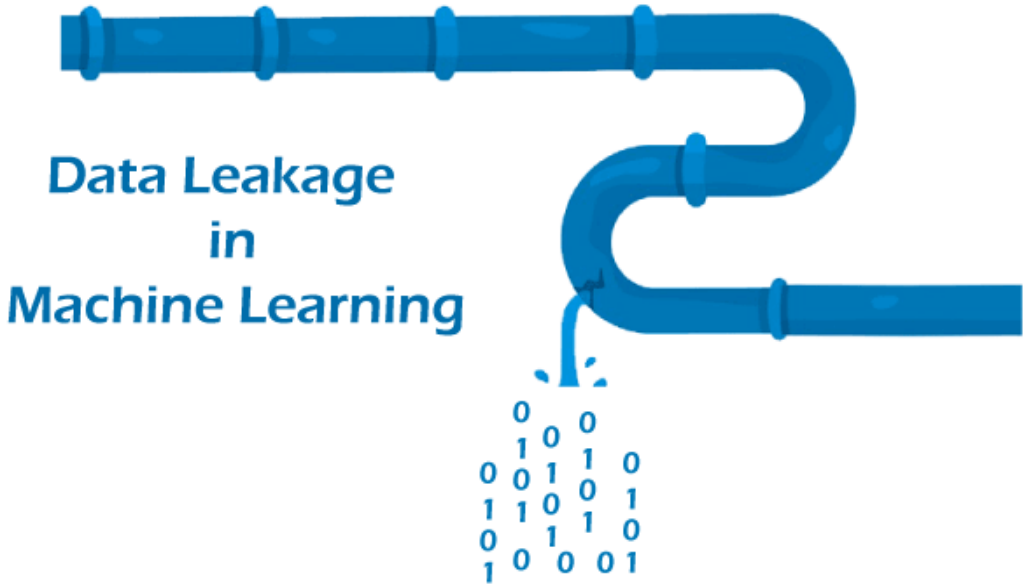
Normalização dos Dados

Pixels entre 0 e 1



Normalização dos Dados

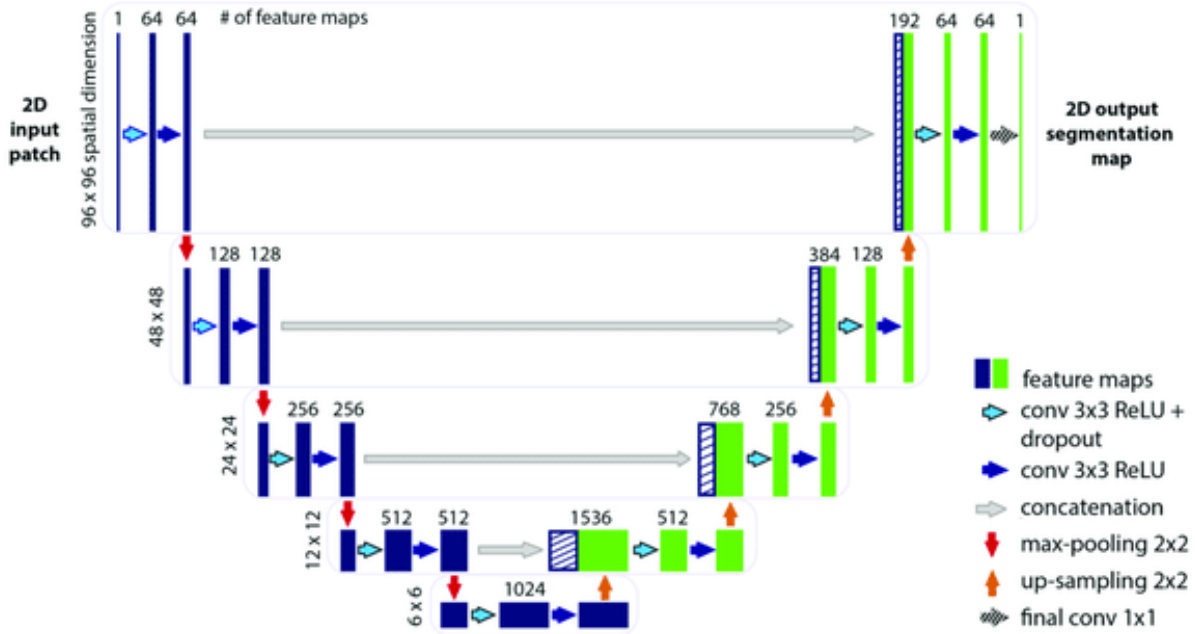
- Data Leakage – Vazamento de dados



Atlântico Avanti

Arquitetura U-NET

Arquitetura U-NET



Arquitetura U-NET

```python
def conv_block(input, num_filters):
    conv = tf.keras.layers.Conv2D(num_filters, 3, padding="same")(input)
    conv = tf.keras.layers.BatchNormalization()(conv)
    conv = tf.keras.layers.Activation("relu")(conv)

    conv = tf.keras.layers.Conv2D(num_filters, 3, padding="same")(conv)
    conv = tf.keras.layers.BatchNormalization()(conv)
    conv = tf.keras.layers.Activation("relu")(conv)
    return conv

def encoder_block(input, num_filters):
    skip = conv_block(input, num_filters)
    pool = tf.keras.layers.MaxPool2D((2,2))(skip)
    return skip, pool

def decoder_block(input, skip, num_filters):
    up_conv = tf.keras.layers.Conv2DTranspose(num_filters, (2,2), strides=2, padding="same")(input)
    conv = tf.keras.layers.Concatenate()([up_conv, skip])
    conv = conv_block(conv, num_filters)
    return conv
```

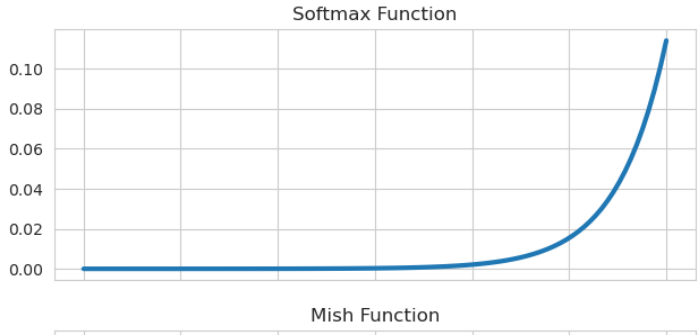
Arquitetura U-NET

```
def Unet(input_shape):
    inputs = tf.keras.layers.Input(input_shape)

    skip1, pool1 = encoder_block(inputs, 64)
    skip2, pool2 = encoder_block(pool1, 128)
    skip3, pool3 = encoder_block(pool2, 256)
    skip4, pool4 = encoder_block(pool3, 512)

    bridge = conv_block(pool4, 1024)

    decode1 = decoder_block(bridge, skip4, 512)
    decode2 = decoder_block(decode1, skip3, 256)
    decode3 = decoder_block(decode2, skip2, 128)
    decode4 = decoder_block(decode3, skip1, 64)
    outputs = tf.keras.layers.Conv2D(1, 1, padding="same", activation="softmax")(decode4)
    model = tf.keras.models.Model(inputs, outputs, name="U-Net")
    return model

unet_model = Unet((512,512,3))
unet_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy', Precision(), Recall(), 'AUC'])
unet_model.summary()
```

Funções de ativação, perda, métricas e otimizador

Função Softmax

$$\phi_i = \frac{e^{z_i}}{\sum_{j \in group} e^{z_j}}$$

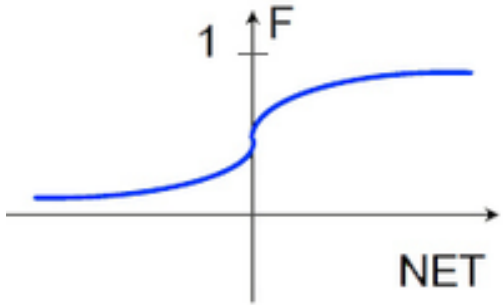


Mudança de função

Função Sigmoide

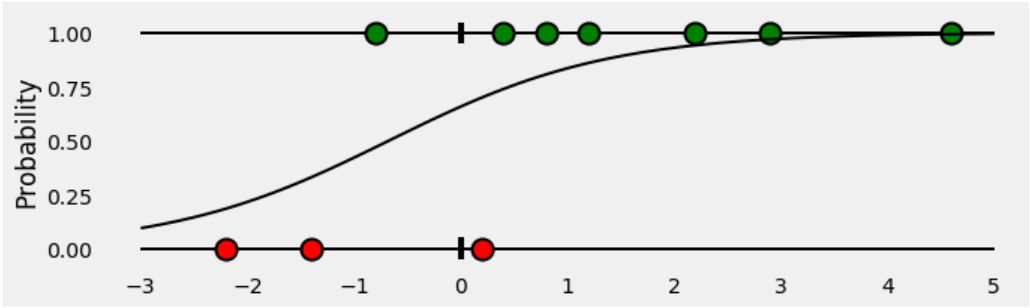
Sigmoide

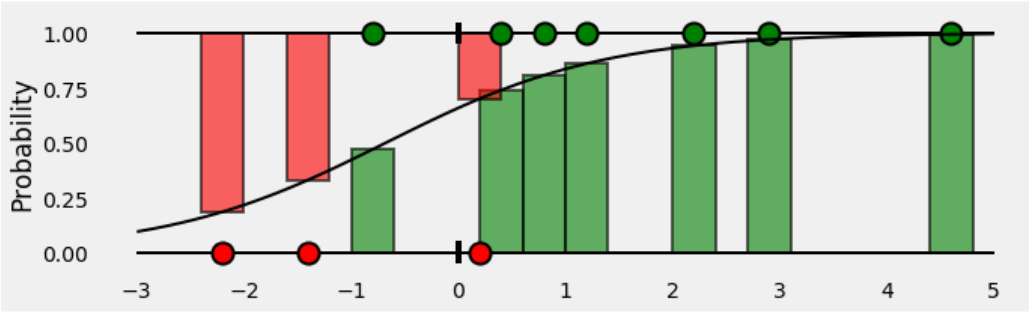
$F = 1 / (1 + e^{-NET})$



Funções de ativação, perda, métricas e otimizador

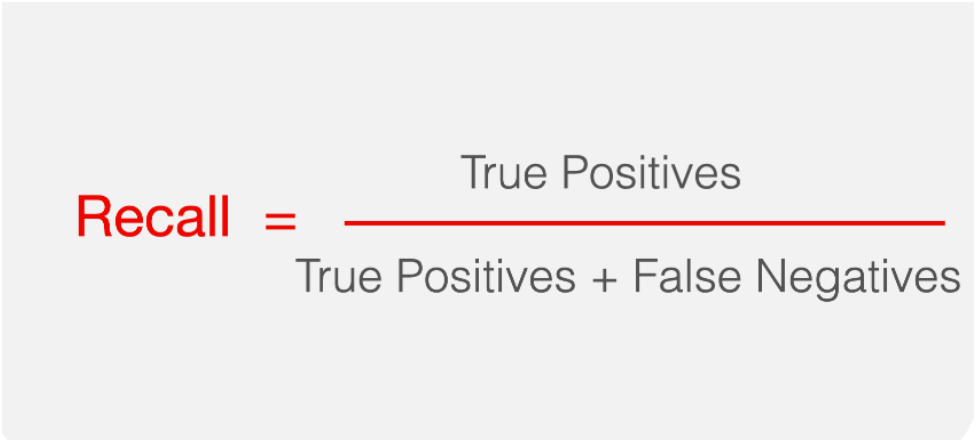
Binary cross-entropy





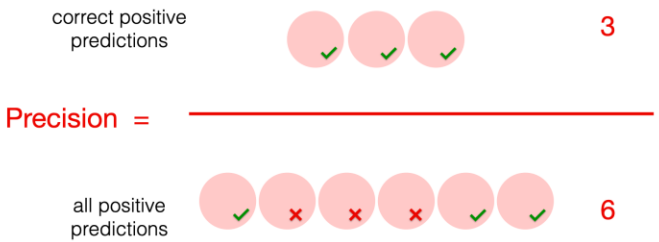
Funções de ativação, perda, métricas e otimizador

Recall



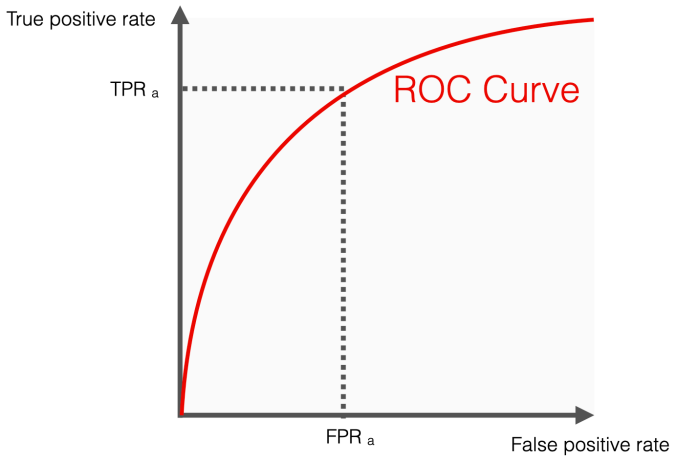
Funções de ativação, perda, métricas e otimizador

Precision

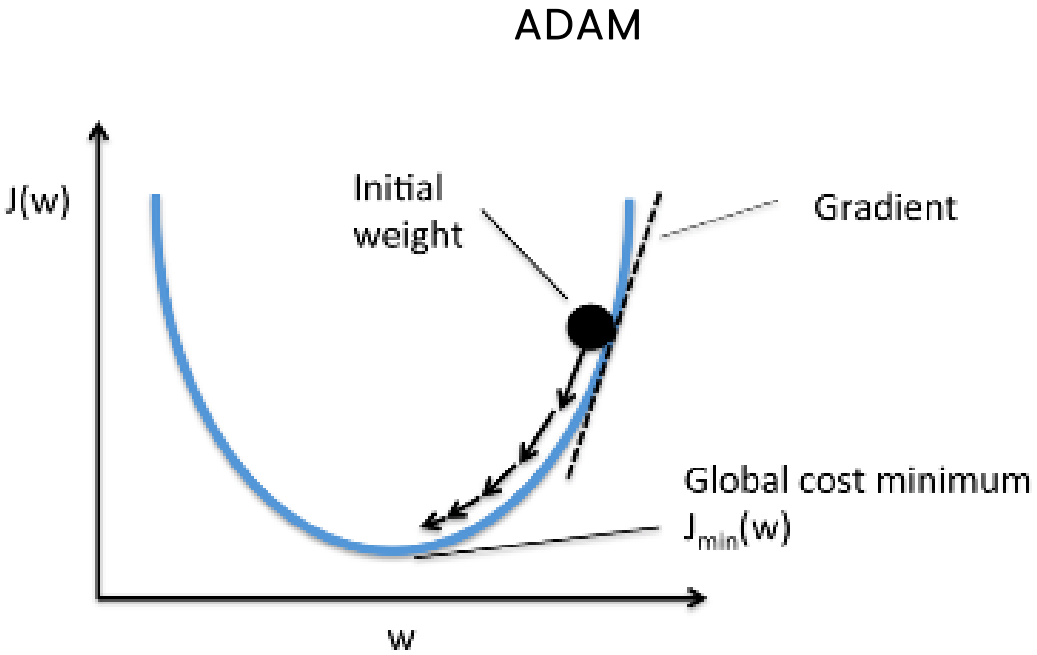


Funções de ativação, perda, métricas e otimizador

AUC



Funções de ativação, perda, métricas e otimizador



Treinamento e validação

Setup - Treinamento e validação

Treinamento

```
unet_result = unet_model.fit(
    image_train, mask_train,
    validation_split = 0.2,
    batch_size = 4,
    epochs = 20)
```

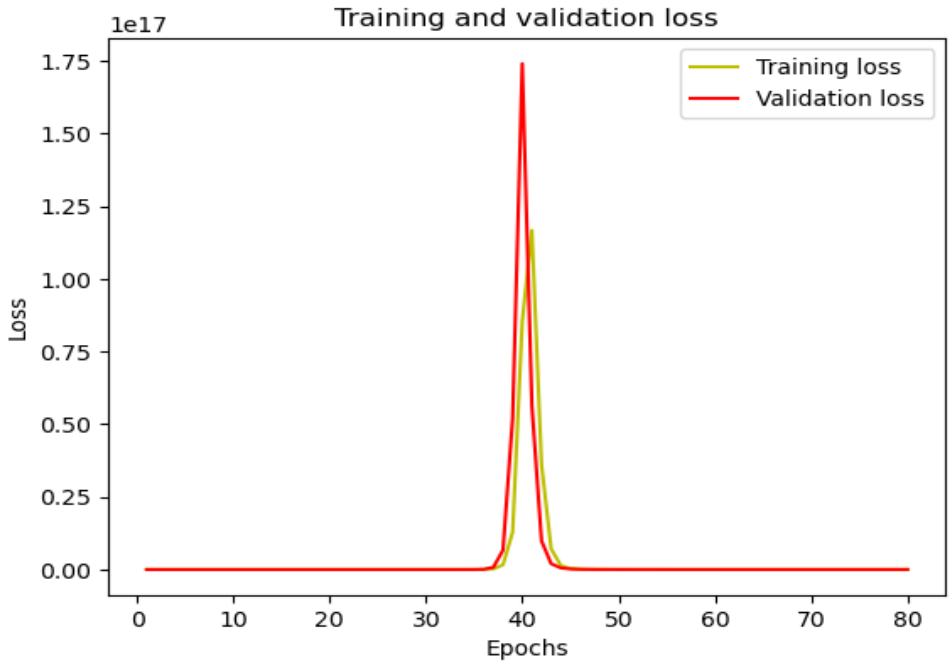
Validação

```
loss = unet_result.history['loss']
val_loss = unet_result.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'y', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

Resultados e conclusões

Resultados - UNET

- 32 de batch size e 80 epochs.
- 74% de accuracy e 81% de val_accuracy: precisão no treino e na validação.
- loss e val_loss: métricas (inadequadas) para o erro no modelo.



Resultados – UNET modificado

- Precision 0.1246 e val_precision 0.1195.
- Recall 1.0000 e val_recall 1.0000.
- Auc 0.5000 e val_auc = 0.5000.
- Overfitting reduzido, loss se manteve problemática.

Resultados – UNET modificado

Problema: loss negativa. Possíveis causas:

- Normalização inadequada do dataset.
- Learning rate alto demais.
- Valores inesperados recebidos pela função loss.

```
train_image_dataset =
np.expand_dims(normalize(np.array(train_image_data
set), axis=1),3)
```

Bynary Cross-Entropy

$$\text{Loss} = -\frac{1}{\text{output size}} \sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

- Calcula a diferença entre a distribuição de probabilidade prevista e a real.
- A variável assume apenas os valores 0 e 1, garantindo que Loss seja sempre maior ou igual a 0.

Bynary Cross-Entropy no Keras

```
bce = target * log(output) + (1 - target) * log(1 - output)
return mean(-bce)
```

- Força as probabilidades preditas a estarem no intervalo (0, 1).
- Se `target` for maior que 1 e `output` for grande o bastante, `bce` seria positivo, logo `mean(-bce)` seria negativo.

Referências Bibliográficas

Referências Bibliográficas

González, Luís. Retinography. Disponível em: Retinography | ICR Ophthalmologic Centre Barcelona (icrcat.com). Acesso em: 10 de novembro de 2023.

Wagih,A. Retina Blood Vessel. Kaggle, 2023. Disponível em: <https://www.kaggle.com/datasets/abdallahwagih/retina-blood-vessel>. Acesso em: 10 de novembro de 2023.

Band, Daniel. Retina Vessel Segmentation with TPU Test (Dice: 0.75). Kaggle, 2023. Disponível em: <https://www.kaggle.com/code/banddaniel/retina-vessel-segmentation-w-tpu-test-dice-0-75/input>. Acesso em: 10 de novembro de 2023.

Yadav, Kiran An In-Depth Exploration of Loss Functions in Deep Learning. Disponível em: <https://www.linkedin.com/pulse/in-depth-exploration-loss-functions-deep-learning-kiran-dev-yadav/>. Acesso em: 10 de novembro de 2023.